SOFTWARE ARCHITECTURE DESIGN

CASE STUDY: RFID BASED TRAIN TICKETING SYSTEM

FUNCTIONAL REQUIRMENTS:

* System will use RFID cards.
* System must allow the commuter to enter the required information to register for the card.
* System must be able to save the registration information in a database.
* System must be able to generate a unique id and pasward for each card.
* System must be able to detect the card on contact from as far as a 100 millimeters.
* System must be able to verify the card’s information from the card’s information in its database.
* System must be able to provide the user proper train station and related offers info.
* System must use a touch screen ticket generator.
* System must be able to verify card’s credit info and automatically deduct the required amount for ticket issued.

QUALITY ATTRIBUTES:

* EFFICIENCY: System will be more efficient than the normal manual ticketing system.
* USEABILITY: system should be easy to use for all age groups with minimal exposure ((up to 30 minutes)
* COST EFFECTIVENESS: system shall reduce the cost of ticketing by 30% by increasing the sales by 25% more than present.
* SECURITY: system must be secure and shall not allow any scams or corruption in financial maters.
* TIME EFFECTIVENESS: the system shall reduce the ticket booking time by negating the long waiting queues.

DIAGRAMS:

1. USECASE DIAGRAM:
2. ANALYSIS CLASS DIAGRAM
3. CRC CARDS
4. SEQUENCE DIAGRAM
5. STATE MACHINE DIAGRAM

USE CASE DIAGRAM:

A close up of a map

Description automatically generated

The use case diagram above shows the two main use cases one being registration of card and the other being the booking of a ticket.

In addition to those are some included and extended events.

The register use case demands entering the commuter info and storing that iofo to system database so both the events are included to registration also the system always generates the password so that also is included. For the ticket booking card verification, entering booking info and payment are all must events so they are included while the presentation of extra offers and the user accepting those offers is a non continuous event hence is extended.

ANALYSIS CLASS DIAGRAM:

A close up of a map

Description automatically generated

The above analysis class diagram contains three entities which are the registration record that contains all information related to commuters data and their pass words, the transport record which is a database containing all the stops and the trains and the offers related to ticket booking and the transaction record that contains the information about the card’s financial details. There are two controller classes , the registration controller that deals with the registration info and also takes part in the verification of a card while booking tickets

And the transaction and ticketing controller that controls all activities like booking the ticket and reserving seats and doing the verification of account details and deducting the funds.

There are three boundary classes. The registration portal allows the commuter a way to register. The card scanner is there to allow card verification and the touch screen handles the rest of the ticketing process by displaying the routes and receiving user choices.

CRC CARDS:

|  |  |
| --- | --- |
| COMMUTER | |
| RESPONSIBILITIES | COLLABORATION |
| * Register for card * Enter user info * Put card on scanner * Insert user id and password * Book ticket | * Registration system * Card scanner * Ticketing system |

|  |  |
| --- | --- |
| REGISTERATION SYSTEM | |
| RESPONSIBILITIES | COLLABORATION |
| * Receive card user info * Save user info in database * Generate user password | * Commuter * Database |

|  |  |
| --- | --- |
| CARD SCANNER | |
| RESPONSIBILITIES | COLLABORATION |
| * Detect card * Receive user id and password * Send user id and password to ticketing system | * Commuter * Ticketing system |

|  |  |
| --- | --- |
| TICKETING SYSTEM | |
| RESPONSIBILITIES | COLLABORATION |
| * Validate user id and password   Receive booking details   * Display ticket details | * Commuter * Database * Card scanner * Transaction manager |

|  |  |
| --- | --- |
| TRANSACTION MANAGER | |
| RESPONSIBILITIES | COLLABORATION |
| * Verify funds presence * Deduct appropriate amount from card * Update funds data in database * Send transaction details to ticketing system | * Database * Ticketing system |

|  |  |
| --- | --- |
| DATABASE | |
| RESPONSIBILITIES | COLLABORATION |
| * Keep registration data * Keep booking info train details and offer details * Keep the financial details of cards | * Registration system * Ticketing system * Transaction manager |

The above CRC cards show the responsibilities and the collaboration classes of our use case’s main classes.

SEQUENCE DIAGRAM:

A close up of a map

Description automatically generated

The above sequence diagram shows the successful flow of registration process of rfid card performed by the commuter.

A close up of a map

Description automatically generated

The above sequence diagram shows the successful ticket booking sequence initiated by the commuter.

STATE MACHINE DIAGRAM:

A close up of a map

Description automatically generated

The above state machine diagram shows different states the system goes through while the commuter books a train ticket.

CLASS SPECIFICATION DIAGRAM:

A close up of a map

Description automatically generated

The above class specification diagram describes all classes their attributes their functions and their relationship to ech other in the RFID train ticketing system.

RELATION OF ALL DIAGRAMS WITH EACH OTHER:

The use case diagram is our basic diagram from which we make our CRC cards and our analysis class diagram using the main classes identified in use case diagram i.e the actors the systems and data records.

The sequence diagram shows the sequence of events that happen in fulfilling a single use case.

While state machine diagram shows different states the system is in while it’s whole process cycle.

Last but not the least the class specification diagram is the final product of the logical design phase that combines all the data gathered in previous models and provides a whole picture of how the system is supposed to look logically